CS 6324: Information Security
Building Key Establishment Protocols

Dr. Junia Valente

Department of Computer Science
The University of Texas at Dallas
Key Establishment Protocols:

We have crypto primitives that give us confidentiality and integrity of messages, but one of the main uses of these primitives are:

1. How can we share a “session” key?

2. How do we make sure we are talking to the person we want? (not an attacker on the network trying to impersonate someone)?
Consider we have parties $A$ and $B$ that want to establish a shared session key.
In general, we assume there is a trusted server on the network between $A$ and $B$ that have a long-term key with each other.

**Security goal**: Party $A$ uses a server called KDC to establish a key $K_{AB}$ that $A$ can use to communicate securely with $B$:

- $K_{AB}$ is known only to $A$ and $B$ (and KDC)
- Moreover, $A$ and $B$ should know who else knows the key
- $A$ and $B$ know that $K_{AB}$ was created by KDC (that they trust)
Attacker Model: Let’s consider this attacker...

- The attacker can be another party in the network that shares a key with the KDC

- Inserts itself in between all communication channels

- Can listen to all messages exchanged
A simple protocol:
- Alice requests to talk to Bob
- KDC generates a key $K_{AB}$ and sends it to Alice and Bob.

Drawbacks:
- KDC has to initiate session with Bob
- Alice may try to communicate with Bob before Bob received the key from the KDC
Key Establishment Protocol: Improvement over initial idea

Have KDC communicate only to Alice
- Alice forwards to Bob information received by KDC

Advantages:
- Reduces communication load on KDC
- Resilient to message delays in network
Key Establishment Protocol: Improvement over initial idea

Attempt #1:

1. \( A \rightarrow S: A, B \) (i.e., A calls S and asks for a key to communicate with B)
2. \( S \rightarrow A: K_{AB} \)
3. \( A \rightarrow B: K_{AB}, A \)

Is this “secure”?
Key Establishment Protocol: Improvement over initial idea

Attempt #1:

1. $A \rightarrow S$: $A, B$ (i.e., A calls S and asks for a key to communicate with B)
2. $S \rightarrow A$: $K_{AB}$
3. $A \rightarrow E$: $K_{AB}, A$
4. $E \rightarrow B$: $K_{AB}, A$

Attacker (Eve) can violate confidentiality goal and learn $K_{AB}$
Alice sends a message to Bob saying to meet her somewhere.

Uh huh.

But Eve sees it, too, and goes to the place.

With you so far.

Bob is delayed, and Alice and Eve meet.

Yeah?

I've discovered a way to get computer scientists to listen to any boring story.
Let’s assume all parties in the network share a secret key $s$ (i.e., $K_s$) with KDC.

**Attempt #2:**

1. $A \rightarrow S$: $A, B$
2. $S \rightarrow A$: $\mathcal{E}_s(K_{AB})$
3. $A \rightarrow B$: $\mathcal{E}_s(K_{AB}), A$

Is this “secure”?

This protocol basically gives away the security key $K_{AB}$
Assumption 1: Attacker can eavesdrop on all the communications

We cannot trust all “n” parties; only the KDC
Kerberos Idea

- When Alice wants to talk to Bob, KDC authenticates the request from Alice, chooses a random key $K$, and sends $\mathcal{E}_{K_A}(K)$, $\mathcal{E}_{K_B}(K)$ to Alice.

- Alice forwards $\mathcal{E}_{K_B}(K)$ to Bob.

- Then, Alice and Bob can use $K$ to communicate.

Note: the KDC can read all communication!
Let’s assume any party $X$ in the system shares a secret key $K_{XS}$ with KDC.

Attempt #3:

1. $A \rightarrow S$: $A$, $B$
2. $S \rightarrow A$: $\mathcal{E}_{K_{AS}}(K_{AB})$, $\mathcal{E}_{K_{BS}}(K_{AB})$
3. $A \rightarrow B$: $\mathcal{E}_{K_{BS}}(K_{AB})$, $A$

Is this “secure”?

The information of who else has the key is not protected.
**Threat Model:**

**Security assumptions**

**Assumption 1:** Attacker can eavesdrop on all the communications

**Assumption 2:** Attacker can intercept and modify all packets in the network

**Assumption 3:** Malicious insiders that share a K with server
Key Establishment Protocol: Man-in-the-middle Attack #1

MITM Attack #1:

1. $A, B$
2. $\varepsilon_{K_{AS}}(K_{AM}), \varepsilon_{K_{MS}}(K_{AM})$
3. $\varepsilon_{K_{MS}}(K_{AM}), A$

Unlike Eve, Mallory "$M$" is an active attacker (e.g., man-in-the-middle attacker) who can modify, substitute messages, replay old messages.

- $M$ changes the intended destination
- $M$ collects the message from $A$ intended for $B$ so that $B$ will not detect any anomaly
- $A$ believes protocol completed successfully while $M$ is masquerading as $B$ and can learn all info $A$ sends.

Problem: key not bound with identity
MITM Attack #2:

1. $A, B$

2. $\varepsilon_{K_{AS}}(K_{AB}), \varepsilon_{K_{BS}}(K_{AB})$

3. $\varepsilon_{K_{BS}}(K_{AB}), A$

3'. $\varepsilon_{K_{BS}}(K_{AB}), C$

Although $M$ does not obtain $K_{AB}$, we still regard this protocol as broken since it does not satisfy our requirement that the users should know who else knows the session key.
Solution:

- Explicitly tell the recipient the expected party that knows the key $K$

- Better yet: bind cryptographically the key $K$, with the identity $B$
  
  e.g., $\mathcal{E}(K, B)$ (authenticated encryption)

Note: For simplicity, we will ignore the message authentication codes exchanged in the next slide, but you should assume that encryption $\mathcal{E}(K, B)$ means it is not only encrypted, but also message authenticated.
First MITM Attack fails:

1. $A, B$

2. $\varepsilon_{K_{AS}}(K_{AM}, M), \varepsilon_{K_{MS}}(K_{AM}, A)$

1'. $A, M$

2'. $\varepsilon_{K_{AS}}(K_{AM}, M), \varepsilon_{K_{MS}}(K_{AM}, A)$

Notice: $M$ cannot change the message. So $A$ rejects key $K_{AM}$.

Now the key is bound with identity
Second MITM Attack fails:

1. $A, B$
2. $\mathcal{E}_{K_{AS}}(K_{AB}, B), \mathcal{E}_{K_{BS}}(K_{AB}, A)$
3. $\mathcal{E}_{K_{BS}}(K_{AB}, A)$

$M$ cannot change the identity in message #3. So, $B$ correctly believes the key is shared with $A$. 

Key Establishment Protocol: Man-in-the-middle Attack #2 - Defeated
Key Establishment Protocol:

Attempt #4:

1. $A \rightarrow S$: $A, B$
2. $S \rightarrow A$: $\mathcal{E}_{K_{AS}}(K, B, \mathcal{E}_{K_{BS}}(K, A))$
3. $A \rightarrow B$: $\mathcal{E}_{K_{BS}}(K, A)$

**Problem**: an attacker can record previous messages exchanges and replay them

Session keys can be compromised (since they are less protected than long-term keys shared with server)
Replay Attack on Attempt #4:

1. $A \rightarrow M: A, B$
2. $M \rightarrow A: \mathcal{E}_{K_{AS}}(K', B, \mathcal{E}_{K_{BS}}(K', A))$
3. $A \rightarrow B: \mathcal{E}_{K_{BS}}(K', A)$

- $M$ uses a previously seen exchange between $A$ and $S$, and manages to convince $A$ and $B$ to use an old key $K'$
- $M$ can replay messages from previous session $K'$
- $M$ can do more cryptanalysis with more messages it sees encrypted with key $K'$
**Freshness:**

- Make sure messages were transmitted recently and are not repeated.

**Weak freshness:** Sender and recipient keep counters

**Strong freshness:** Nonce, Clocks
Freshness:

- To prevent replay, we need to add something that distinguishes one session from another

- **General paradigm**: (1) server sends a “challenge” and (2) user provides a correct “response”

- Challenge should (at least) be non-repeating
  - Counter (weak freshness: prevents replay but not “delay” attacks)
  - Random value (nonce)
  - Time
Attempt #5:

1. $A \rightarrow S$: $A, B, N_A$

2. $S \rightarrow A$: $\mathcal{E}_{K_{AS}}(K, B, N_A, \mathcal{E}_{K_{BS}}(K, A))$

3. $A \rightarrow B$: $\mathcal{E}_{K_{BS}}(K, A)$

4. $B \rightarrow A$: $\mathcal{E}_K(N_B)$

5. $A \rightarrow B$: $\mathcal{E}_K(N_B - 1)$

Freshness achieved with nonces $N_A$ and $N_B$
Assumption 1: Attacker can eavesdrop on all the communications
Assumption 2: Attacker can intercept and modify all packets in the network
Assumption 3: Malicious insiders that share a K with server
Assumption 4: Attacker can get a session key from an old key establishment
**Key Establishment Protocol:**

*Needham–Schroeder protocol (1978)*

*Best practice: assume attacker can obtain a session key $K'$ used in any sufficiently old previous run of the protocol*

**Problem:** replay attack: if $M$ learns of an old key $K$, then $B$ cannot tell the key is not fresh.

1. $A \rightarrow B$: $\mathcal{E}_{K_B^S}(K, A)$
2. $B \rightarrow A$: $\mathcal{E}_K(N_B)$
3. $A \rightarrow B$: $\mathcal{E}_K(N_B - 1)$

*Replay Attack using Old Session Key:*

1. $M \rightarrow B$: $\mathcal{E}_{K_B^S}(K', A)$
2. $B \rightarrow M$: $\mathcal{E}_{K'}(N_B)$
3. $M \rightarrow B$: $\mathcal{E}_{K'}(N_B - 1)$
Attempt #6:

Solution 1: Submit nonces from both users to $S$

\[ 1. B, N_B \]
\[ A \] \[ B \] \[ 2. A, B, N_A, N_B \] \[ 3. \epsilon_{K_{AS}}(K_{AB}, B, N_A, \epsilon_{K_{BS}}(K_{AB}, A, N_B)) \] \[ S \] \[ 4. \epsilon_{K_{BS}}(K_{AB}, A, N_B) \]

This attempt is secure! (Finally!)
Attempt #7:

Solution 2: use synchronized clocks and timestamps as nonce

This attempt is secure too!

Minimizes messages: Instead of nonces, use timestamps (assumes all nodes in the system have a synchronized clock).

A and B reject any message that is too old.
- Use encryption and authentication for messages transmitted

- When distributing a key, bind the key and user identity cryptographically (i.e., say specifically who the key is meant for)

- Ensure messages are fresh